

INICIAR SESIÓN

NUESTROS PLANES

TODOS LOS  
CURSOS

FORMACIONES

CURSOS

PARA  
EMPRESAS

ARTÍCULOS DE TECNOLOGÍA &gt; PROGRAMACIÓN

# Entendiendo el Lazy y el Eager Load de JPA



felipe-oliveira

14/07/2021

Uno de los puntos más importantes a analizar en un software es el desempeño y optimizar al máximo las queries SQL, puede resultar en una ganancia de desempeño considerable. Teniendo esto en cuenta, veamos cómo funcionan Lazy y Eager Load de JPA (Java Persistence API), porque su uso incorrecto provoca una gran pérdida en el desempeño de una aplicación, lo que obliga a la base de datos a queries innecesarias.

Cuando modelamos un sistema usando la orientación a objetos, creamos varias relaciones que pueden ser @OneToOne, @ManyToOne, @OneToMany o @ManyToMany y para la base de datos, cada relación es una nueva tabla a consultar. Supongamos que tenemos dos entidades, Alumno y Matrícula:

```
@Entity
public class Alumno {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nombre;
    @OneToOne
    private Matricula matricula;

    // getters y setters omitidos
```

```
}  
@Entity  
public class Matricula {  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
    @Column(unique = true, nullable = false)  
    private String codigo;  
  
    // getters y setters omitidos  
  
}
```

Por estándar, cuando la relación se anota con `@OneToOne` o `@ManyToOne`, se carga en modo Eager, es decir, cuando hicimos algún tipo de búsqueda en la entidad como por ejemplo un `find(Alumno.class, 1)`, se cargará junto con la entidad, en cuyo caso JPA ejecutará una única query. En el ejemplo anterior colocamos la anotación `@OneToOne` en el atributo `Matricula`, ¿tiene sentido? No, pues generalmente un alumno tiene varias matrículas, así que cambiemos la anotación de la relación de clase `Alumno` a `@OneToMany` y el tipo de `List`:

```
@Entity  
public class Alumno {  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
    private String nombre;  
    @OneToMany  
    private List<Matricula> matriculas;  
  
    // getters y setters omitidos  
  
}  
@Entity  
public class Matricula {  
    @Id
```

```
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
@Column(unique = true, nullable = false)
private String codigo;

// getters y setters omitidos

}
```

Por estándar, cuando la relación está anotada con `@OneToMany` o `@ManyToMany`, se carga en modo Lazy, es decir, cuando hicimos algún tipo de búsqueda en la entidad como por ejemplo `find(Alumno.class, 1)`, no se cargará con la entidad, solo cuando ejecutemos el comando `getMatriculas()` se cargará la relación. Por el modo Lazy, solo cargamos informaciones cuando ejecutamos un getter para hacer un `aluno.getMatriculas().getCodigo()` dentro de un for para obtener el código de todas las matrículas del alumno puede traer problema de desempeño a la aplicación, ya que JPA ejecutará varias queries.

```
for(Alumno alumno : alumnos) {
    for(Matricula matricula : alumno.getMatriculas()){
        System.out.println(matricula.getCodigo());
    }
}
```

El código desde dentro del for ejecutará el tamaño de la lista veces, por ejemplo, si el tamaño de la lista es N, el código se ejecutará N veces, luego la JPA ejecutará N queries en la base de datos, también podemos contar la query para cargar la entidad. Este problema se conoce como consultas  $[N + 1]$ , uno de los problemas que deben evitarse al utilizar JPA e Hibernate.

Además del problema mencionado anteriormente, debemos tener cuidado con `LazyInitializationException`.

Como vimos en esta publicación, cargar las entidades incorrectamente puede causar pérdida de desempeño, así que recuerde considerar la mejor estrategia para cargar sus relaciones.

Puedes leer también:

- [JPA con Hibernate: Herencia y Mapeos](#)
- [Reciba notificaciones de la api de Servlet a través de Listeners](#)
- [Conozca la API de fechas de Java 8](#)

ARTÍCULOS DE TECNOLOGÍA > PROGRAMACIÓN

**En Alura encontrarás variados cursos sobre Programación.  
¡Comienza ahora!**

**SEMESTRAL**

**US\$49,90**

un solo pago de US\$49,90

- ✓ 218 cursos
- ✓ Videos y actividades 100% en Español
- ✓ Certificado de participación
- ✓ Estudia las 24 horas, los 7 días de la semana
- ✓ Foro y comunidad exclusiva para resolver tus dudas
- ✓ Acceso a todo el contenido de la plataforma por 6 meses

**¡QUIERO EMPEZAR A ESTUDIAR!**

[Paga en moneda local en los siguientes países](#)

**ANUAL**

**US\$79,90**

un solo pago de US\$79,90

- ✓ 218 cursos
- ✓ Videos y actividades 100% en Español
- ✓ Certificado de participación
- ✓ Estudia las 24 horas, los 7 días de la semana
- ✓ Foro y comunidad exclusiva para resolver tus dudas
- ✓ Acceso a todo el contenido de la plataforma por 12 meses

**¡QUIERO EMPEZAR A ESTUDIAR!**

[Paga en moneda local en los siguientes países](#)

Acceso a todos  
los cursos

Estudia las 24 horas,  
dónde y cuándo quieras

Nuevos cursos  
cada semana

## NAVEGACIÓN

PLANES

INSTRUCTORES

BLOG

POLÍTICA DE PRIVACIDAD

TÉRMINOS DE USO

SOBRE NOSOTROS

PREGUNTAS FRECUENTES

## ¡CONTÁCTANOS!

¡QUIERO ENTRAR EN CONTACTO!

## BLOG

PROGRAMACIÓN

FRONT END

DATA SCIENCE

INNOVACIÓN Y GESTIÓN

DEVOPS

AOVS Sistemas de Informática S.A

CNPJ 05.555.382/0001-33

## SÍGUENOS EN NUESTRAS REDES SOCIALES



## ALIADOS



En Alura somos unas de las Scale-Ups seleccionadas por Endeavor, programa de aceleración de las empresas que más crecen en el país.



Fuimos unas de las 7 startups seleccionadas por Google For Startups en participar del programa Growth Academy en 2021

POWERED BY

## CURSOS

### Cursos de Programación

Lógica de Programación | Java

### Cursos de Front End

HTML y CSS | JavaScript | React

### Cursos de Data Science

Data Science | Machine Learning | Excel | Base de Datos | Data Visualization | Estadística

### Cursos de DevOps

Docker | Linux

### Cursos de Innovación y Gestión

Productividad y Calidad de Vida | Transformación Ágil | Marketing Analytics |  
Liderazgo y Gestión de Equipos | Startups y Emprendimiento